Практическое занятие № 16

Тема: Разработка многооконного приложения для работы с одотабличной БД в IDE PyCharm Community.

Постановка первой задачи: Приложение АБИТУРИЕНТ для автоматизации работы приемной комиссии,
которая обеспечивает обработку анкетных данных абитуриентов. Таблица Анкета
содержит следующие данные об абитуриентах: Регистрационный номер, Фамилия, Имя,
Отчество, Дата Рождения, Награды (наличие кр. Диплома или медали (да/нет)), Адрес,
выбранная Специальность.
БД должна обеспечивать вывод анкетных данных по фамилии.

Текст первой программы:

```
#Приложение АБИТУРИЕНТ для автоматизации работы приемной комиссии,
#которая обеспечивает обработку анкетных данных абитуриентов. Таблица Анкета
#содержит следующие данные об абитуриентах: Регистрационный номер, Фамилия,
Имя,
#Отчество, Дата Рождения, Награды (наличие кр. Диплома или медали (да/нет)),
Адрес,
#выбранная Специальность.
#БД должна обеспечивать вывод анкетных данных по фамилии.

import tkinter as tk
from tkinter import ttk
import sqlite3 as sq


class Main(tk.Frame):

    """Класс для главного окна"""

    def __init__(self, root):
        super().__init__(root)
        self.init_main()
        self.db = db
        self.view_records()

    def init_main(self):
        toolbar = tk.Frame(bg='#c8a2c8', bd=4)
        toolbar.pack(side=tk.TOP, fill=tk.X)

        self.add_img = tk.PhotoImage(file="gif/11.gif")
        self.btn_open_dialog = tk.Button(toolbar, text='Добавить
абитуриента', command=self.open_dialog, bg='#e6a8d7', bd=0,
                                         compound=tk.TOP, image=self.add_img,
width=158, height=40)
        self.btn_open_dialog.pack(side=tk.LEFT)

        self.update_img = tk.PhotoImage(file="gif/12.gif")
        self.btn_edit_dialog = tk.Button(toolbar, text="Редактировать",
command=self.open_update_dialog, bg='#c8a2c8',
                                         bd=0, compound=tk.TOP,
image=self.update_img, width=158, height=40)
        self.btn_edit_dialog.pack(side=tk.LEFT)

        self.delete_img = tk.PhotoImage(file="gif/13.gif")
        self.btn_delete = tk.Button(toolbar, text="Удалить запись",
```

```
command=self.delete_records, bg='#c8a2c8',
                                    bd=0, compound=tk.TOP,
image=self.delete_img, width=158, height=40)
        self.btn_delete.pack(side=tk.LEFT)

        self.search_img = tk.PhotoImage(file="gif/14.gif")
        self.btn_search = tk.Button(toolbar, text="Поиск записи",
command=self.open_search_dialog, bg='#c8a2c8',
                                bd=0, compound=tk.TOP, image=self.search_img,
width=158, height=40)
        self.btn_search.pack(side=tk.LEFT)

        self.refresh_img = tk.PhotoImage(file="gif/15.gif")
        self.btn_refresh = tk.Button(toolbar, text="Обновить экран",
command=self.view_records, bg='#c8a2c8',
                                bd=0, compound=tk.TOP, image=self.refresh_img,
width=158, height=40)
        self.btn_refresh.pack(side=tk.LEFT)

        self.tree = ttk.Treeview(self, columns=('id', 'ab_f', 'ab_i', 'ab_o',
'date', 'awards', 'adres', 'spec'), height=19, show='headings')

        self.tree.column('id', width=46, anchor=tk.CENTER)
        self.tree.column('ab_f', width=100, anchor=tk.CENTER)
        self.tree.column('ab_i', width=100, anchor=tk.CENTER)
        self.tree.column('ab_o', width=100, anchor=tk.CENTER)
        self.tree.column('date', width=100, anchor=tk.CENTER)
        self.tree.column('awards', width=100, anchor=tk.CENTER)
        self.tree.column('adres', width=100, anchor=tk.CENTER)
        self.tree.column('spec', width=100, anchor=tk.CENTER)

        self.tree.heading('id', text='Номер')
        self.tree.heading('ab_f', text='Фамилия')
        self.tree.heading('ab_i', text='Имя')
        self.tree.heading('ab_o', text='Отчество')
        self.tree.heading('date', text='Дата рождения')
        self.tree.heading('awards', text='Награды')
        self.tree.heading('adres', text='Адрес')
        self.tree.heading('spec', text='Специальность')

        self.tree.pack()

    def records(self, id, ab_f, ab_i, ab_o, date, awards, adres, spec):
        self.db.insert_data(id, ab_f, ab_i, ab_o, date, awards, adres, spec)
        self.view_records()

    def update_record(self, id, ab_f, ab_i, ab_o, date, awards, adres, spec):
        self.db.cur.execute("""UPDATE abiturient SET id=?, ab_f=?, ab_i=?,
ab_o=?, date=?, awards=?, adres=?, spec=? WHERE id=?""",
                            (id, ab_f, ab_i, ab_o, date, awards, adres, spec,
self.tree.set(self.tree.selection()[0], '#1')))
        self.db.con.commit()
        self.view_records()

    def view_records(self):
        self.db.cur.execute("""SELECT * FROM abiturient""")
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in
self.db.cur.fetchall()]

    def delete_records(self):
```

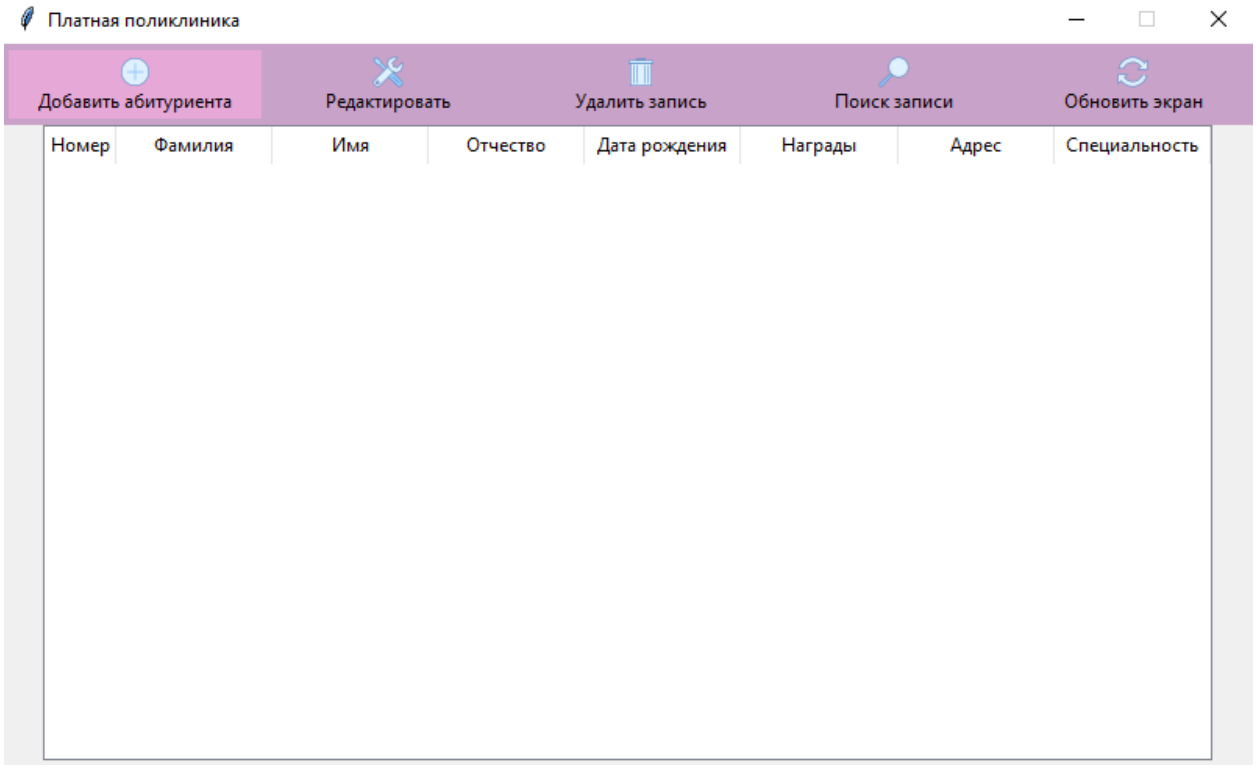
```
        for selection_item in self.tree.selection():
            self.db.cur.execute("""DELETE FROM abiturient WHERE id=?""",
(self.tree.set(selection_item, '#1'),))
        self.db.con.commit()
        self.view_records()

    def search_records(self, ab_f):
        ab_f = (ab_f,)
        self.db.cur.execute("""SELECT * FROM abiturient WHERE ab_f=?""",
ab_f)
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in
self.db.cur.fetchall()]


    def open_dialog(self):
        Child(root, app)

    def open_update_dialog(self):
        Update()

    def open_search_dialog(self):
        Search()

class Child(tk.Toplevel):

    """Класс для дочернего окна"""

    def __init__(self, root, app):
        super().__init__(root)
        self.init_child()
        self.view = app

    def init_child(self):
        self.title('Добавить абитуриента')
        self.geometry('498x300+580+300')
        self.resizable(False, False)

        label_id = tk.Label(self, text='ID')
        label_id.place(x=50, y=25)
        self.entry_id = ttk.Entry(self, width=33)
        self.entry_id.place(x=175, y=25)

        label_ab_f = tk.Label(self, text='Фамилия')
        label_ab_f.place(x=50, y=50)
        self.entry_ab_f = ttk.Entry(self, width=33)
        self.entry_ab_f.place(x=175, y=50)

        label_ab_i = tk.Label(self, text='Имя')
        label_ab_i.place(x=50, y=75)
        self.entry_ab_i = ttk.Entry(self, width=33)
        self.entry_ab_i.place(x=175, y=75)

        label_ab_o = tk.Label(self, text='Отчество')
        label_ab_o.place(x=50, y=100)
        self.entry_ab_o = ttk.Entry(self, width=33)
        self.entry_ab_o.place(x=175, y=100)

        label_date = tk.Label(self, text='Дата Рождения')
        label_date.place(x=50, y=125)
        self.entry_date = ttk.Entry(self, width=33)
```

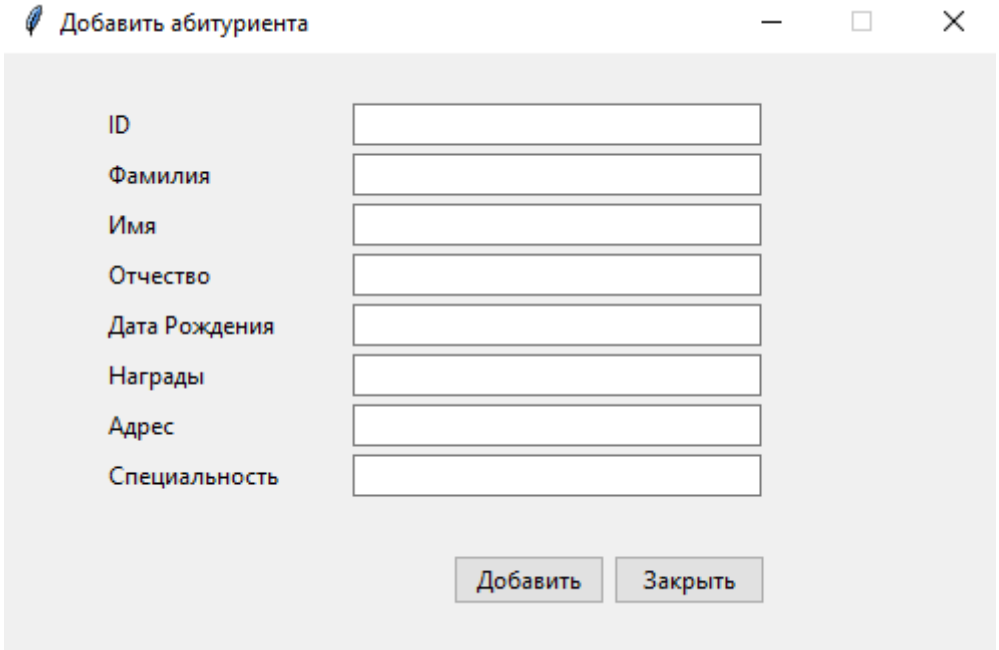
```
        self.entry_date.place(x=175, y=125)

        label_awards = tk.Label(self, text='Награды')
        label_awards.place(x=50, y=150)
        self.entry_awards = ttk.Entry(self, width=33)
        self.entry_awards.place(x=175, y=150)

        label_adres = tk.Label(self, text='Адрес')
        label_adres.place(x=50, y=175)
        self.entry_adres = ttk.Entry(self, width=33)
        self.entry_adres.place(x=175, y=175)

        label_spec = tk.Label(self, text='Специальность')
        label_spec.place(x=50, y=200)
        self.entry_spec = ttk.Entry(self, width=33)
        self.entry_spec.place(x=175, y=200)

        btn_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)
        btn_cancel.place(x=305, y=250)

        self.btn_ok = ttk.Button(self, text='Добавить')
        self.btn_ok.place(x=225, y=250)
        self.btn_ok.bind('<Button-1>', lambda event:
self.view.records(self.entry_id.get(),

self.entry_ab_f.get(),

self.entry_ab_i.get(),

self.entry_ab_o.get(),

self.entry_date.get(),

self.entry_awards.get(),

self.entry_adres.get(),

self.entry_spec.get()))

        self.grab_set()
        self.focus_set()

class Update(Child):
    def __init__(self):
        super().__init__(root, app)
        self.init_edit()
        self.view = app

    def init_edit(self):
        self.title("Редактировать запись")
        btn_edit = ttk.Button(self, text="Редактировать")
        btn_edit.place(x=210, y=250)
        btn_edit.bind('<Button-1>', lambda event:
self.view.update_record(self.entry_id.get(),

self.entry_ab_f.get(),

self.entry_ab_i.get(),

self.entry_ab_o.get(),
```

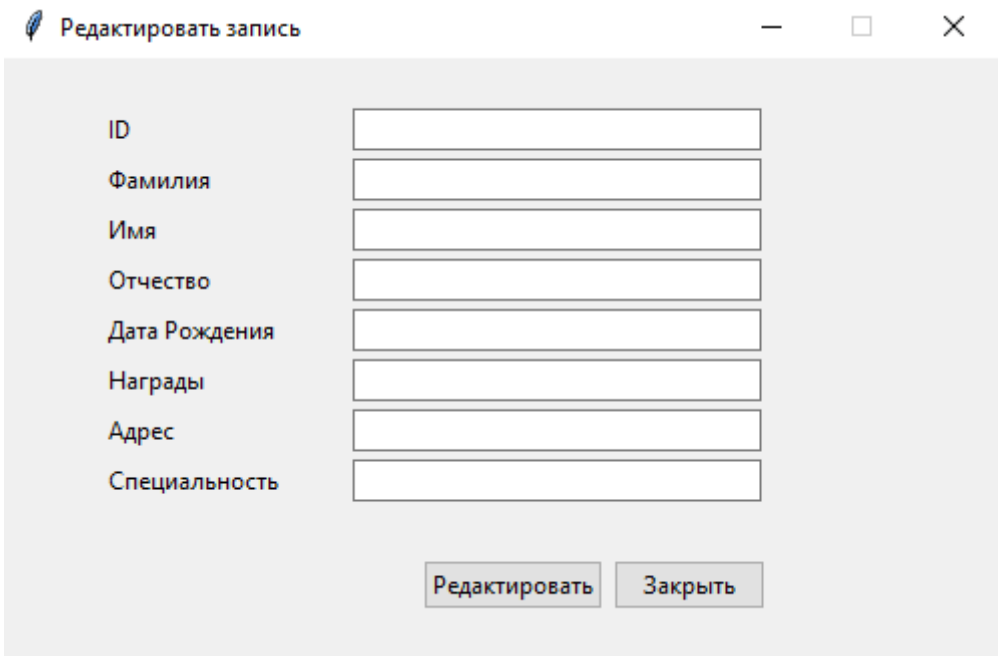
```
self.entry_date.get(),

self.entry_awards.get(),

self.entry_adres.get(),

self.entry_spec.get()))
        self.btn_ok.destroy()

class Search(tk.Toplevel):
    def __init__(self):
        super().__init__()
        self.init_search()
        self.view = app

    def init_search(self):
        self.title("Поиск")
        self.geometry("430x100+550+350")
        self.resizable(False, False)

        label_search = tk.Label(self, text="Поиск (найдет записи по
фамилии)")
        label_search.place(x=50, y=20)

        self.entry_search = ttk.Entry(self)
        self.entry_search.place(x=255, y=20, width=155)

        btn_cancel = ttk.Button(self, text="Закрыть", command=self.destroy)
        btn_cancel.place(x=335, y=50)

        btn_search = ttk.Button(self, text="Поиск")
        btn_search.place(x=255, y=50)
        btn_search.bind('<Button-1>', lambda event:
self.view.search_records(self.entry_search.get()))
        btn_search.bind('<Button-1>', lambda event: self.destroy(), add='+')

class DB:
    def __init__(self):

        with sq.connect('abiturient.db') as self.con:
            self.cur = self.con.cursor()
            self.cur.execute("""CREATE TABLE IF NOT EXISTS abiturient (
                    id INTEGER,
                    ab_f TEXT NOT NULL,
                    ab_i TEXT NOT NULL,
                    ab_o TEXT NOT NULL,
                    date INTEGER,
                    awards TEXT NOT NULL,
                    adres TEXT NOT NULL,
                    spec TEXT NOT NULL
                )""")

    def insert_data(self, id, ab_f, ab_i, ab_o, date, awards, adres, spec):
        self.cur.execute("""INSERT INTO abiturient(id, ab_f, ab_i, ab_o,
date, awards, adres, spec) VALUES (?, ?, ?, ?, ?, ?, ?, ?)""",
                         (id, ab_f, ab_i, ab_o, date, awards, adres,
spec))
        self.con.commit()

if __name__ == "__main__":
    root = tk.Tk()
```

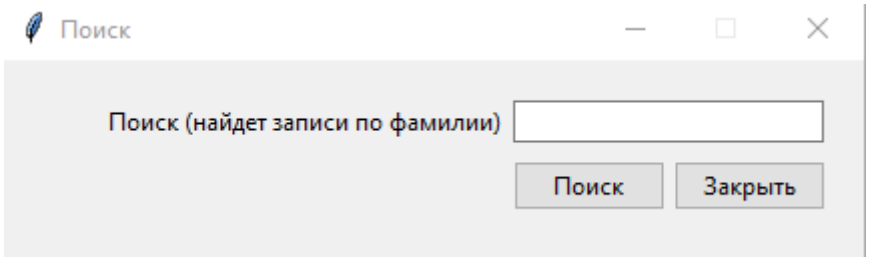
```
db = DB()
app = Main(root)
app.pack()
root.title("Платная поликлиника")
root.geometry("802x463+350+175")
root.resizable(False, False)
root.mainloop()
```

Протокол работы первой программы:









Вывод: в процессе выполнения практического занятия выработал навыки разработки многооконного приложения для работы с одотабличной БД в IDE PyCharm Community.